window

Nov 12, 2019 23:27

Supported Clients

SmartClient WebClient NGClient

Methods Summary

ToolBar	addToolBar(window, name)	Creates and returns a toolbar for a specific window.
ToolBar	addToolBar(window, name, row)	Creates and returns a toolbar for a specific window.
ToolBar	addToolBar(window, name, displayname)	Creates and returns a toolbar for a specific window.
ToolBar	addToolBar(window, name, displayname, row)	Creates and returns a toolbar for a specific window.
ToolBar	addToolBar(name)	Add a toolbar.
ToolBar	addToolBar(name, row)	Add a toolbar.
ToolBar	addToolBar(name, displayname)	Add a toolbar.
ToolBar	addToolBar(name, displayname, row)	Add a toolbar.
void	cancelFormPopup()	Close the current form popup panel without assigning a value to the configured data provider.
void	closeFormPopup(retval)	Close the current form popup panel and assign the value to the configured data provider.
Popup	createPopupMenu()	Creates a new popup menu that can be populated with items and displayed.
Boolean	createShortcut(shortcut, methodName)	Create a shortcut.
Boolean	createShortcut(shortcut, methodName, arguments)	Create a shortcut.
Boolean	createShortcut(shortcut, methodName, contextFilter)	Create a shortcut.
Boolean	createShortcut(shortcut, method, contextFilter, arguments)	Create a shortcut.
Boolean	createShortcut(shortcut, methodName, contextFilter, arguments, consumeEvent)	Create a shortcut.
Boolean	createShortcut(shortcut, method)	Create a shortcut.
Boolean	createShortcut(shortcut, method, arguments)	Create a shortcut.
Boolean	createShortcut(shortcut, method, contextFilter)	Create a shortcut.
Boolean	createShortcut(shortcut, method, contextFilter, arguments)	Create a shortcut.
Boolean	createShortcut(shortcut, method, contextFilter, arguments, consumeEvent)	Create a shortcut.
MenuBar	getMenuBar()	Get the menubar of the main window, or of a named window.
MenuBar	getMenuBar(windowName)	Get the menubar of the main window, or of a named window.
ToolBar	getToolBar(window, name)	Get the toolbar of a specific window from the toolbar panel by name.
ToolBar	getToolBar(name)	Get the toolbar from the toolbar panel by name.
Array	getToolbarNames()	Get all toolbar names from the toolbar panel.
Array	getToolbarNames(window)	Get all toolbar names from the toolbar panel of a specific window.
void	maximize()	Maximize the current window or the window with the specified name (Smart client only).
void	maximize(windowName)	Maximize the current window or the window with the specified name (Smart client only).
Boolean	removeShortcut(shortcut)	Remove a shortcut.
Boolean	removeShortcut(shortcut, contextFilter)	Remove a shortcut.
void	removeToolBar(window, name)	Remove the toolbar from the toolbar panel of a specific window.
void	removeToolBar(name)	Remove the toolbar from the toolbar panel.
void	setFullScreen(full)	Bring the window into/out of fullsceen mode.
void	setStatusBarVisible(visible)	Show or hide the statusbar.
void	setToolBarAreaVisible(visible)	Show or hide the toolbar area.
void	showFormPopup(elementToShowRelatedTo, form, scope, dataproviderID)	Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the specified scope.
void	showFormPopup(elementToShowRelatedTo, form, scope, dataproviderID, width, height)	Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the specified scope.
void	showFormPopup(elementToShowRelatedTo, form, scope, dataproviderID, width, height, x, y)	Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the specified scope.

Methods Details

addToolBar(window, name)

```
Creates and returns a toolbar for a specific window.
```

Parameters

Object window ;
String name the name by which this toolbar is identified in code. If display name is missing, name will be used as displayName as well.

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method addToolBar only works in the smart client.

// create a window
var win = application.createWindow("myWindow", JSWindow.WINDOW);

// add a toolbar with only a name
var toolbar0 = plugins.window.addToolBar(win,"toolbar_0");
toolbar0.addButton("click me 0", callback_function);

// add a toolbar with a name and the row you want it to show at
// row number starts at 0
var toolbar1 = plugins.window.addToolBar(win,"toolbar_1", 2);
toolbar1.addButton("click me 1", callback_function);

// add a toolbar with a name and display name
var toolbar2 = plugins.window.addToolBar(win,"toolbar_2", "toolbar_2_internal_name");
toolbar2.addButton("click me 2", callback_function);

// add a toolbar with a name, display name and the row you want the
// toolbar to show at. row number starts at 0
var toolbar3 = plugins.window.addToolBar(win,"toolbar_3", "toolbar_3_internal_name", 3);
toolbar3.addButton("click me 3", callback_function);

win.show(forms.Myform)
```

addToolBar(window, name, row)

Creates and returns a toolbar for a specific window.

Parameters

Object	window	;
String	name	the name by which this toolbar is identified in code. If display name is missing, name will be used as displayName as well.
Number	row	the row inside the toolbar panel where this toolbar is to be added.

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method addToolBar only works in the smart client.

// create a window
var win = application.createWindow("myWindow", JSWindow.WINDOW);

// add a toolbar with only a name
var toolbar0 = plugins.window.addToolBar(win,"toolbar_0");
toolbar0.addButton("click me 0", callback_function);

// add a toolbar with a name and the row you want it to show at
// row number starts at 0
var toolbar1 = plugins.window.addToolBar(win,"toolbar_1", 2);
toolbar1.addButton("click me 1", callback_function);

// add a toolbar with a name and display name
var toolbar2 = plugins.window.addToolBar(win,"toolbar_2", "toolbar_2_internal_name");
toolbar2.addButton("click me 2", callback_function);

// add a toolbar with a name, display name and the row you want the
// toolbar to show at. row number starts at 0
var toolbar3 = plugins.window.addToolBar(win,"toolbar_3", "toolbar_3_internal_name", 3);
toolbar3.addButton("click me 3", callback_function);

win.show(forms.Myform)
```

addToolBar(window, name, displayname)

Creates and returns a toolbar for a specific window.

Parameters

Object	window	;
String	name	the name by which this toolbar is identified in code
String	displayname	the name by which this toolbar will be identified in the UI. (for example in the toolbar panel's context menu)

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method addToolBar only works in the smart client.

// create a window
var win = application.createWindow("myWindow", JSWindow.WINDOW);

// add a toolbar with only a name
var toolbar0 = plugins.window.addToolBar(win,"toolbar_0");
toolbar0.addButton("click me 0", callback_function);

// add a toolbar with a name and the row you want it to show at
// row number starts at 0
var toolbar1 = plugins.window.addToolBar(win,"toolbar_1", 2);
toolbar1.addButton("click me 1", callback_function);

// add a toolbar with a name and display name
var toolbar2 = plugins.window.addToolBar(win,"toolbar_2", "toolbar_2_internal_name");
toolbar2.addButton("click me 2", callback_function);

// add a toolbar with a name, display name and the row you want the
// toolbar to show at. row number starts at 0
var toolbar3 = plugins.window.addToolBar(win,"toolbar_3", "toolbar_3_internal_name", 3);
toolbar3.addButton("click me 3", callback_function);

win.show(forms.Myform)
```

addToolBar(window, name, displayname, row)

Creates and returns a toolbar for a specific window.

Parameters

Object	window	;
String	name	the name by which this toolbar is identified in code.
String	displayname	the name by which this toolbar will be identified in the UI. (for example in the toolbar panel's context menu)
Number	row	the row inside the toolbar panel where this toolbar is to be added.

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method addToolBar only works in the smart client.

// create a window
var win = application.createWindow("myWindow", JSWindow.WINDOW);

// add a toolbar with only a name
var toolbar0 = plugins.window.addToolBar(win,"toolbar_0");
toolbar0.addButton("click me 0", callback_function);

// add a toolbar with a name and the row you want it to show at
// row number starts at 0
var toolbar1 = plugins.window.addToolBar(win,"toolbar_1", 2);
toolbar1.addButton("click me 1", callback_function);

// add a toolbar with a name and display name
var toolbar2 = plugins.window.addToolBar(win,"toolbar_2", "toolbar_2_internal_name");
toolbar2.addButton("click me 2", callback_function);

// add a toolbar with a name, display name and the row you want the
// toolbar to show at. row number starts at 0
var toolbar3 = plugins.window.addToolBar(win,"toolbar_3", "toolbar_3_internal_name", 3);
toolbar3.addButton("click me 3", callback_function);

win.show(forms.Myform)
```

addToolBar(name)

Add a toolbar.

Parameters

String	name	the name by which this toolbar is identified in code. If display name is missing, name will be used as displayName as well.

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method addToolBar only works in the smart client.

// add a toolbar with only a name
var toolbar0 = plugins.window.addToolBar("toolbar_0");
toolbar0.addButton("click me 0", feedback_button);

// add a toolbar with a name and the row you want it to show at
// row number starts at 0
var toolbar1 = plugins.window.addToolBar("toolbar_1", 2);
toolbar1.addButton("click me 1", feedback_button);

// add a toolbar with a name and display name
var toolbar2 = plugins.window.addToolBar("toolbar_2", "toolbar_2_internal_name");
toolbar2.addButton("click me 2", feedback_button);

// add a toolbar with a name, display name and the row you want the
// toolbar to show at. row number starts at 0
var toolbar3 = plugins.window.addToolBar("toolbar_3", "toolbar_3_internal_name", 3);
toolbar3.addButton("click me 3", feedback_button);
```

addToolBar(name, row)

Add a toolbar.

Parameters

String name the name by which this toolbar is identified in code. If display name is missing, name will be used as displayName as well.
Number row the row inside the toolbar panel where this toolbar is to be added.

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method addToolBar only works in the smart client.

// add a toolbar with only a name
var toolbar0 = plugins.window.addToolBar("toolbar_0");
toolbar0.addButton("click me 0", feedback_button);

// add a toolbar with a name and the row you want it to show at
// row number starts at 0
var toolbar1 = plugins.window.addToolBar("toolbar_1", 2);
toolbar1.addButton("click me 1", feedback_button);

// add a toolbar with a name and display name
var toolbar2 = plugins.window.addToolBar("toolbar_2", "toolbar_2_internal_name");
toolbar2.addButton("click me 2", feedback_button);

// add a toolbar with a name, display name and the row you want the
// toolbar to show at. row number starts at 0
var toolbar3 = plugins.window.addToolBar("toolbar_3", "toolbar_3_internal_name", 3);
toolbar3.addButton("click me 3", feedback_button);
```

addToolBar(name, displayname)

Add a toolbar.

Parameters

String name the name by which this toolbar is identified in code. If display name is missing, name will be used as displayName as well.
String displayname the name by which this toolbar will be identified in the UI. (for example in the toolbar panel's context menu)

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method addToolBar only works in the smart client.

// add a toolbar with only a name
var toolbar0 = plugins.window.addToolBar("toolbar_0");
toolbar0.addButton("click me 0", feedback_button);

// add a toolbar with a name and the row you want it to show at
// row number starts at 0
var toolbar1 = plugins.window.addToolBar("toolbar_1", 2);
toolbar1.addButton("click me 1", feedback_button);

// add a toolbar with a name and display name
var toolbar2 = plugins.window.addToolBar("toolbar_2", "toolbar_2_internal_name");
toolbar2.addButton("click me 2", feedback_button);

// add a toolbar with a name, display name and the row you want the
// toolbar to show at. row number starts at 0
var toolbar3 = plugins.window.addToolBar("toolbar_3", "toolbar_3_internal_name", 3);
toolbar3.addButton("click me 3", feedback_button);
```

addToolBar(name, displayname, row)

Add a toolbar.

Parameters

String	name	the name by which this toolbar is identified in code. If display name is missing, name will be used as displayName as well.
String	displayname	the name by which this toolbar will be identified in the UI. (for example in the toolbar panel's context menu)
Number	row	the row inside the toolbar panel where this toolbar is to be added.

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method addToolBar only works in the smart client.

// add a toolbar with only a name
var toolbar0 = plugins.window.addToolBar("toolbar_0");
toolbar0.addButton("click me 0", feedback_button);

// add a toolbar with a name and the row you want it to show at
// row number starts at 0
var toolbar1 = plugins.window.addToolBar("toolbar_1", 2);
toolbar1.addButton("click me 1", feedback_button);

// add a toolbar with a name and display name
var toolbar2 = plugins.window.addToolBar("toolbar_2", "toolbar_2_internal_name");
toolbar2.addButton("click me 2", feedback_button);

// add a toolbar with a name, display name and the row you want the
// toolbar to show at. row number starts at 0
var toolbar3 = plugins.window.addToolBar("toolbar_3", "toolbar_3_internal_name", 3);
toolbar3.addButton("click me 3", feedback_button);
```

cancelFormPopup()

Close the current form popup panel without assigning a value to the configured data provider.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the
specified scope.
plugins.window.showFormPopup(null,forms.orderPicker,foundset.getSelectedRecord(),"order_id");
// do call closeFormPopup(ordervalue) from the orderPicker form
```

closeFormPopup(retval)

Close the current form popup panel and assign the value to the configured data provider.

Parameters

Object retval return value for data provider

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the
specified scope.
plugins.window.showFormPopup(null,forms.orderPicker,foundset.getSelectedRecord(),"order_id");
// do call closeFormPopup(ordervalue) from the orderPicker form
```

createPopupMenu()

Creates a new popup menu that can be populated with items and displayed.

Returns

Popup

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// create a popup menu
var menu = plugins.window.createPopupMenu();
// add a menu item
menu.addMenuItem("an entry", feedback);

if (event.getSource()) {
        // display the popup over the component which is the source of the event
        menu.show(event.getSource());
        // display the popup over the components, at specified coordinates relative to the component
        //menu.show(event.getSource(), 10, 10);
        // display the popup at specified coordinates relative to the main window
        //menu.show(100, 100);
}
```

createShortcut(shortcut, methodName)

Create a shortcut.

Parameters

String shortcut ;
String methodName scopes.scopename.methodname or formname.methodname String to target the method to execute

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

createShortcut(shortcut, methodName, arguments)

Create a shortcut.

Parameters

String shortcut ;
String methodName scopes.scopename.methodname or formname.methodname String to target the method to execute
Array arguments ;

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

createShortcut(shortcut, methodName, contextFilter)

Create a shortcut.

Parameters

String shortcut ;
String methodName scopes.scopename.methodname or formname.methodname String to target the method to execute
String contextFilter form or element name (ng only - specified by formName.elementName); only triggers the shortcut when on this form/element

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

createShortcut(shortcut, method, contextFilter, arguments)

Create a shortcut.

Parameters

String	shortcut	;
Object	method	the method/function that needs to be called when the shortcut is hit
String	contextFilter	form or element name (ng only - specified by formName.elementName); only triggers the shortcut when on this form/element
Array	arguments	;

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

createShortcut(shortcut, methodName, contextFilter, arguments, consumeEvent)

Create a shortcut.

Parameters

String	shortcut	;
String	methodName	scopes.scopename.methodname or formname.methodname String to target the method to execute
String	contextFilter	form or element name (ng only - specified by formName.elementName); only triggers the shortcut when on this form/element
Array	arguments	;
Boolean	consumeEvent	if true then the shotcut will consume the event and the default browser behavior will not be executed (default false)

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

createShortcut(shortcut, method)

Create a shortcut.

Parameters

String shortcut ;
Function method the method/function that needs to be called when the shortcut is hit

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

createShortcut(shortcut, method, arguments)

Create a shortcut.

Parameters

String	shortcut	;
Function	method	the method/function that needs to be called when the shortcut is hit
Array	arguments	;

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

createShortcut(shortcut, method, contextFilter)

Create a shortcut.

Parameters

String	shortcut	;
Function	method	the method/function that needs to be called when the shortcut is hit
String	contextFilter	form or element name (ng only - specified by formName.elementName); only triggers the shortcut when on this form/element

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

createShortcut(shortcut, method, contextFilter, arguments)

Create a shortcut.

Parameters

String	shortcut	;
Function	method	the method/function that needs to be called when the shortcut is hit
String	contextFilter	form or element name (ng only - specified by formName.elementName); only triggers the shortcut when on this form/element
Array	arguments	;

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

createShortcut(shortcut, method, contextFilter, arguments, consumeEvent)

Create a shortcut.

Parameters

String	shortcut	;
Function	method	the method/function that needs to be called when the shortcut is hit
String	contextFilter	form or element name (ng only - specified by formName.elementName); only triggers the shortcut when on this form/element
Array	arguments	;
Boolean	consumeEvent	if true then the shotcut will consume the event and the default browser behavior will not be executed (default false)

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

getMenuBar()

Get the menubar of the main window, or of a named window.

Returns

MenuBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// create a new window
var win = application.createWindow("windowName", JSWindow.WINDOW);
// show a form in the new window
forms.my_form.controller.show(win);
// retrieve the menubar of the new window
var menubar = plugins.window.getMenuBar("windowName");
// add a new menu to the menubar, with an item in it
var menu = menubar.addMenu();
menu.text = "New Menu";
menu.addMenuItem("an entry", feedback);
// retrieve the menubar of the main window
var mainMenubar = plugins.window.getMenuBar();
// add a new menu to the menubar of the main window
var menuMain = mainMenubar.addMenu();
menuMain.text = "New Menu in Main Menubar";
menuMain.addMenuItem("another entry", feedback);
```

getMenuBar(windowName)

Get the menubar of the main window, or of a named window.

Parameters

String windowName the name of the window

Returns

MenuBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// create a new window
var win = application.createWindow("windowName", JSWindow.WINDOW);
// show a form in the new window
forms.my_form.controller.show(win);
// retrieve the menubar of the new window
var menubar = plugins.window.getMenuBar("windowName");
// add a new menu to the menubar, with an item in it
var menu = menubar.addMenu();
menu.text = "New Menu";
menu.addMenuItem("an entry", feedback);
// retrieve the menubar of the main window
var mainMenubar = plugins.window.getMenuBar();
// add a new menu to the menubar of the main window
var menuMain = mainMenubar.addMenu();
menuMain.text = "New Menu in Main Menubar";
menuMain.addMenuItem("another entry", feedback);
```

getToolBar(window, name)

Get the toolbar of a specific window from the toolbar panel by name.

Parameters

Object window ;
String name ;

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method getToolBar only works in the smart client.

// create a window
        var win = application.createWindow("myWindow", JSWindow.WINDOW);
// the toolbar must first be created with a call to addToolbar
plugins.window.addToolBar(win,"toolbar_0");

// show the empty toolbar and wait 4 seconds
win.show(forms.MyForm)
application.updateUI(4000)

// get the toolbar at the panel by name
var toolbar = plugins.window.getToolBar(win,"toolbar_0");
// add a button to the toolbar
toolbar.addButton("button", callback_function);
```

getToolBar(name)

Get the toolbar from the toolbar panel by name.

Parameters

String name ;

Returns

ToolBar

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method getToolBar only works in the smart client.

// the toolbar must first be created with a call to addToolbar
plugins.window.addToolBar("toolbar_0");

// get the toolbar at the panel by name
var toolbar = plugins.window.getToolBar("toolbar_0");
// add a button to the toolbar
toolbar.addButton("button", feedback_button);
```

getToolbarNames()

Get all toolbar names from the toolbar panel.

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method getToolbarNames only works in the smart client.

// create an array of toolbar names
var names = plugins.window.getToolbarNames();

// create an empty message variable
var message = "";

// loop through the array
for (var i = 0 ; i < names.length ; i++) {
        //add the name(s) to the message
        message += names[i] + "\n";
}

// show the message
plugins.dialogs.showInfoDialog("toolbar names", message);
```

getToolbarNames(window)

Get all toolbar names from the toolbar panel of a specific window.

Parameters

Object window ;

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method getToolbarNames only works in the smart client.
// create a window
        var win = application.createWindow("myWindow", JSWindow.WINDOW);
// the toolbar must first be created with a call to addToolbar
         plugins.window.addToolBar(win,"toolbar_0");
  plugins.window.addToolBar(win,"toolbar_1");
// create an array of toolbar names
var names = plugins.window.getToolbarNames(win);

// create an empty message variable
var message = "";

// loop through the array
for (var i = 0 ; i < names.length ; i++) {
        //add the name(s) to the message
        message += names[i] + "\n";
}

// show the message
plugins.dialogs.showInfoDialog("toolbar names", message);
```

maximize()

Maximize the current window or the window with the specified name (Smart client only).

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// maximize the main window:
plugins.window.maximize();

// create a new window
var win = application.createWindow("windowName", JSWindow.WINDOW);
// show a form in the new window
forms.my_form.controller.show(win);
// maximize the window
plugins.window.maximize("windowName");
```

maximize(windowName)

Maximize the current window or the window with the specified name (Smart client only).

Parameters

String windowName ;

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// maximize the main window:
plugins.window.maximize();

// create a new window
var win = application.createWindow("windowName", JSWindow.WINDOW);
// show a form in the new window
forms.my_form.controller.show(win);
// maximize the window
plugins.window.maximize("windowName");
```

removeShortcut(shortcut)

Remove a shortcut.

Parameters

String shortcut ;

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

removeShortcut(shortcut, contextFilter)

Remove a shortcut.

Parameters

String shortcut ;
String contextFilter form or element name (ng only - specified by formName.elementName); only triggers the shortcut when on this form/element

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// this plugin uses the java keystroke parser
// see http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/KeyStroke.html#getKeyStroke(java.lang.String)
// global handler
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut);
// global handler with a form context filter
plugins.window.createShortcut('control shift I', scopes.globals.handleOrdersShortcut, 'frm_contacts');
// form method called when shortcut is used
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut);
// form method called when shortcut is used and arguments are passed to the method
plugins.window.createShortcut('control RIGHT', forms.frm_contacts.handleMyShortcut, new Array(argument1,
argument2));
// Passing the method argument as a string prevents unnecessary form loading
//plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', new Array(argument1,
argument2));
// Passing the method as a name and the contextFilter set so that this shortcut only trigger on the form
'frm_contacts'.
plugins.window.createShortcut('control RIGHT', 'frm_contacts.handleMyShortcut', 'frm_contacts', new Array
(argument1, argument2));
// Num Lock and Substract shortcuts
plugins.window.createShortcut("NUMPAD8", handleMyShortcut);
plugins.window.createShortcut("SUBTRACT", handleMyShortcut);
// remove global shortcut and form-level shortcut
plugins.window.removeShortcut('menu 1');
plugins.window.removeShortcut('control RIGHT', 'frm_contacts');
// consuming they keystroke so that a default browser event will not happen
plugins.window.createShortcut('F4', scopes.globals.handleOrdersShortcut, 'frm_contacts', null, true);
// shortcut handlers are called with an JSEvent argument
///
// * Handle keyboard shortcut.
//
// * @param {JSEvent} event the event that triggered the action
// */
//function handleShortcut(event)
//{
//  application.output(event.getType()) // returns 'menu 1'
//  application.output(event.getFormName()) // returns 'frm_contacts'
//  application.output(event.getElementName()) // returns 'contact_name_field' or null when no element is
selected
//}
// NOTES:
// 1) shortcuts will not override existing operating system or browser shortcuts,
// choose your shortcuts carefully to make sure they work in all clients.
// 2) always use lower-case letters for modifiers (shift, control, etc.), otherwise createShortcut will fail.
```

removeToolBar(window, name)

Remove the toolbar from the toolbar panel of a specific window.

Parameters

Object window ;
String name ;

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method removeToolBar only works in the smart client.
// create a window
        var win = application.createWindow("myWindow", JSWindow.WINDOW);
// the toolbar must first be created with a call to addToolbar
var toolbar = plugins.window.addToolBar(win,"toolbar_0");

// add a button to the toolbar
toolbar.addButton("button", callcbackMethod);

// show the toolbar with the button and wait 4 seconds, then remove it
win.show(forms.MyForm)
application.updateUI(4000)

// removing a toolbar from the toolbar panel is done by name
// the plugin checks the existence of the toolbar
// when the toolbar does not exist it will not throw an error though.
plugins.window.removeToolBar(win,"toolbar_0");
```

removeToolBar(name)

Remove the toolbar from the toolbar panel.

Parameters

String name ;

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Note: method removeToolBar only works in the smart client.

// the toolbar must first be created with a call to addToolbar
var toolbar = plugins.window.addToolBar("toolbar_0");

// add a button to the toolbar
toolbar.addButton("button", feedback_button);

// removing a toolbar from the toolbar panel is done by name
// the plugin checks the existence of the toolbar
// when the toolbar does not exist it will not throw an error though.
plugins.window.removeToolBar("toolbar_0");
```

setFullScreen(full)

Bring the window into/out of fullsceen mode.

Parameters

Boolean full ;

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// active fullscreen mode
plugins.window.setFullScreen(true);
```

setStatusBarVisible(visible)

Show or hide the statusbar.

Parameters

Boolean visible ;

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// hide the statusbar
plugins.window.setStatusBarVisible(false);
```

setToolBarAreaVisible(visible)

Show or hide the toolbar area.

Parameters

Boolean visible ;

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// hide the toolbar area
plugins.window.setToolBarAreaVisible(false);
```

showFormPopup(elementToShowRelatedTo, form, scope, dataproviderID)

Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the specified scope.

Parameters

Object	elementToShowRelatedTo	element to show related to or null to center in screen
Object	form	the form to show
Object	scope	the scope to put retval into
String	dataproviderID	the dataprovider of scope to fill

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the
specified scope.
plugins.window.showFormPopup(null,forms.orderPicker,foundset.getSelectedRecord(),"order_id");
// do call closeFormPopup(ordervalue) from the orderPicker form
```

showFormPopup(elementToShowRelatedTo, form, scope, dataproviderID, width, height)

Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the specified scope.

Parameters

Object	elementToShowRelatedTo	element to show related to or null to center in screen
Object	form	the form to show
Object	scope	the scope to put retval into
String	dataproviderID	the dataprovider of scope to fill
Number	width	popup width
Number	height	popup height

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the
specified scope.
plugins.window.showFormPopup(null,forms.orderPicker,foundset.getSelectedRecord(),"order_id");
// do call closeFormPopup(ordervalue) from the orderPicker form
```

showFormPopup(elementToShowRelatedTo, form, scope, dataproviderID, width, height, x, y)

Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the specified scope. Can show relative to a component or at specified coordinates.
Show on specified location is only supported in NGClient.

Parameters

Object	elementToShowRelatedTo	element to show related to or null to center in screen
Object	form	the form to show
Object	scope	the scope to put retval into
String	dataproviderID	the dataprovider of scope to fill
Number	width	popup width
Number	height	popup height
Number	x	popup x location
Number	y	popup y location

Supported Clients

NGClient

Sample

```
// Show a form as popup panel, where the closeFormPopup can pass return a value to a dataprovider in the
specified scope.
plugins.window.showFormPopup(null,forms.orderPicker,foundset.getSelectedRecord(),"order_id");
// plugins.window.showFormPopup(null,forms.orderPicker,foundset.getSelectedRecord(),"order_id",-1,-1,100,100);
// do call closeFormPopup(ordervalue) from the orderPicker form
```